I Claim:

1. A computer apparatus suitable for use in the combined compilation and verification of platform neutral bytecode instructions resulting in optimized machine code, comprising:

a central processing unit (CPU);

a computer memory coupled to said CPU, said computer memory comprised of a computer readable medium;

a compilation-verification program embodied on said computer readable medium, said compilation-verification program comprising:

a first code segment that receives a bytecode listing;

a second code segment that verifies said bytecode listing is free of malicious and improper code and compiles said bytecode listing into machine code; and

a third code segment that interprets and executes said machine code.

2. A computer apparatus as recited in Claim 1 wherein said computer program simultaneously verifies and compiles said bytecode listing into optimized machine code.

3. A computer apparatus suitable for use in the combined compilation and verification of platform neutral bytecode instructions resulting in optimized machine code, comprising:

a development or target computer system, said development or target computer system comprised of a computer readable storage medium containing a compilation verification program and one or more class files, said one or more class files containing one or more methods containing bytecode instruction listings;

said compilation-verification program contained on said storage medium comprised of a first plurality of subset instructions, said first plurality configured to execute verification of said bytecode instruction listings;

said compilation-verification program contained on said storage medium further comprised of a second plurality of subset instructions, said second plurality configured to execute compilation of said bytecode instruction listings; and

an optimized machine code simultaneously resulting from said first and second subset instructions.

4. A computer apparatus as recited in Claim 3 wherein said first plurality of subset instructions evaluates said bytecode instructions to detect improper data types and improper stack usage.

5. A computer apparatus as recited in Claim 3 wherein said second plurality of subset instructions evaluates said bytecode instructions for complete compilation of said bytecode instructions into said optimized machine code.

6. A computer apparatus as recited in Claim 3 wherein said first and second plurality of subset instructions are executed simultaneously.

7. A computer implemented method for facilitating combined compilation and verification of platform neutral bytecode instructions resulting in optimized machine code, comprising the steps of:

receiving a class file onto a computer readable medium containing compilation procedure instructions, said class file containing one or more methods containing platform neutral bytecode listings;

executing said compilation procedure instructions on said bytecode listings, said compilation procedure instructions also simultaneously verifying said bytecode listings; and

producing verified optimized machine code on said computer readable medium.

8. A computer implemented method as recited in Claim 7 wherein said compilation procedure creates storage for each bytecode instruction to store stack status and marks.

9. A computer implemented method as recited in Claim 8 wherein said compilation procedure creates storage to store actual types of stack values and local variables.

10. A computer implemented method as recited in Claim 9 wherein said compilation procedure initializes stack status of the first bytecode instruction to empty.

11. A computer implemented method as recited in Claim 10 wherein said compilation procedure initializes stack status of exception handler target instructions to contain a given exception object.

12. A computer implemented method as recited in Claim 11 wherein said compilation procedure sets marks of said first bytecode instruction and handler target instructions to setup.

13. A computer implemented method as recited in Claim 12 wherein said compilation procedure sets all other marks to none.

14. A computer implemented method as recited in Claim 13 wherein said compilation procedure initializes actual local variable types from method signature.

15. A computer implemented method as recited in Claim 14 wherein said compilation procedure sets said first bytecode instruction to be the actual instruction.

16. A computer implemented method as recited in Claim 15 wherein said compilation procedure repeats until there are no more instructions marked as setup.

17. A computer implemented method as recited in Claim 16 wherein said compilation procedure determines if said actual instruction is not marked as setup and if not marked as setup then:

selecting the next instruction in the bytecode marked as setup as said actual instruction; and

loading actual stack and local variable types from the stack map in bytecode belonging to said actual instruction.

18. A computer implemented method as recited in Claim 17 wherein said compilation procedure determines if said actual instruction is in the scope of an exception handler and if said actual instruction is in the scope then:

verify compatibility between actual local variable types and stack map for the exception handler entry in bytecode.

19. A computer implemented method as recited in Claim 18 wherein said compilation procedure sets the mark of selected instruction to handled.

20. A computer implemented method as recited in Claim 19 wherein said compilation procedure copies stack status of actual instruction to new stack status.

21. A computer implemented method as recited in Claim 20 wherein said compilation procedure determines if said actual instruction pops one or more values from the stack and if said actual instruction pops one or more values from said stack then:

verify compatibility between the stack status and types and the values expected by said actual instruction; and

modify new stack status according to said actual instruction.

22. A computer implemented method as recited in Claim 21 wherein said compilation procedure determines if said actual instruction pushes one or more values to the stack and if said actual instruction pushes one or more values to the stack then:

modify new stack status according to said actual instruction; and

set new actual stack types according to said actual instruction.

23. A computer implemented method as recited in Claim 22 wherein said compilation procedure determines if said actual instruction reads a local variable and if said actual instruction reads said local variable then:

verify compatibility between actual local variable types and said actual instruction.

24. A computer implemented method as recited in Claim 23 wherein said compilation procedure determines if said actual instruction writes to a local variable and if said actual instruction writes to said local variable then:

modify actual local variable types according to said actual instruction.

25. A computer implemented method as recited in Claim 24 wherein said compilation procedure determines if a successor instruction is immediately following said actual instruction and if said successor instruction is not immediately following said actual instruction then:

if said successor instruction is marked as none, initialize the stack status of said successor instruction to the new stack status and mark said successor instruction as setup;

verify compatibility between new stack status and stack map for said successor instruction in the bytecode; and

verify compatibility between actual stack and local variable types and stack map for said successor instruction in the bytecode.

26. A computer implemented method as recited in Claim 25 wherein said compilation procedure determines if an instruction immediately following said actual instruction is a successor of said actual instruction and if said following instruction is a successor of said actual instruction then:

if said successor instruction is marked as none, initialize the stack status of said following instruction to the new stack status and mark said following instruction as setup;

if there is a stack map in the bytecode for said following instruction, verify compatibility between new stack status and stack map for said successor instruction in the bytecode; and

verify compatibility between actual stack and local variable types and stack map for said successor instruction in the bytecode.

27. A computer implemented method as recited in Claim 26 wherein said compilation procedure changes said actual instruction to the immediately following instruction.

28. A computer implemented method as recited in Claim 27 wherein said compilation procedure repeats for each said method.

29. A computer implemented method as recited in Claim 28 wherein said compilation procedure repeats for each said class.